CHAPTER 1

PROGRAM 1

Design, Develop and Implement a menu driven Program in C for the following Array operations

a. **Creating an Array of N Integer Elements**
b. **Display of Array Elements with Suitable Headings**
c. **Inserting an Element (ELEM) at a given valid Position (POS)**
d. **Deleting an Element at a given valid Position (POS)**
e. **Exit.**

Support the program with functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
int a[20];
int n, val, i, pos, choice;
/*Function Prototype*/
void create();
void display();
void insert();
void delet();
int main()
{
      while(1)
      {
      printf("\n\n--------MENU----------\n");
      printf("1.CREATE\n");
      printf("2.DISPLAY\n");
      printf("3.INSERT\n");
      printf("4.DELETE\n");
      printf("5.EXIT\n");
      printf("-----------------------");
      printf("\nENTER YOUR CHOICE:\t");
      scanf("%d",&choice); //Reading user choice
     switch(choice)
        {
            case 1: create();          //to create an array
                    break;
            case 2: if(n)
                      display();   //to display the elements
                    else
                       {
                            printf("Array is empty\n");
                       }
                    break;
            case 3: :insert();   // to insert elements
                    break;

            case 4:if(n)
```

```
        delet(); //to delete an item at specific position
        else
                        {
                                printf("Array is empty\n");
                        }
                        break;
                case 5: exit(0);          //exit the program
                        break;
                default:printf("\nInvalid choice:\n");
                        break;
        }
    }
    return 0;
}
    //creating an array
    void create()
    {
        printf("\nEnter the size of the array elements:\t");
        scanf("%d",&n);
        printf("\nEnter the elements for the array:\n");
        for(i=1;i<=n;i++)
        {
         scanf("%d",&a[i]);
        }
    }
    //displaying an array elements
    void display()
    {
        int i;
        printf("\nThe array elements are:\n");
        for(i=1;i<=n;i++)
        {
          printf("%d\t",a[i]);
        }
    }
    //inserting an element into an array
    void insert()
    {
        printf("\nEnter the position for the new element:\t");
        scanf("%d",&pos);
        if(pos>(n+1))
        printf("\n Invalid Position\n");
        else
        {
                printf("\nEnter the element to be inserted :\t");
                scanf("%d",&val);
                for(i=n;i>=pos;i--)
                {
```

```
                        a[i+1]=a[i];
                }
                a[pos]=val;
                n=n+1;
        }
}
//deleting an array element
void delet()
{
        printf("\nEnter the position of the element to be deleted:\t");
        scanf("%d",&pos);
        if(pos>n)
       printf("Invalid Position\n");
        else
        {
                val=a[pos];
                for(i=pos;i<n;i++)
                {
                        a[i]=a[i+1];
                }
                n=n-1;
                printf("\nThe deleted element is =%d",val);
        }
}
```

OUTPUT:

```
--------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      1

Enter the size of the array elements:   4

Enter the elements for the array:
20 5 10 30


--------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      2

The array elements are:
20      5       10      30

--------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      3

Enter the position for the new element: 1

Enter the element to be inserted :      90
--------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      2

The array elements are:
90      20      5       10      30
```

```
--------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      3

Enter the position for the new element: 9

 Invalid Position


--------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      3

Enter the position for the new element: 6

Enter the element to be inserted :      60


--------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      2

The array elements are:
90      20      5       10      30      60
ENTER YOUR CHOICE:      4

Enter the position of the element to be deleted:        1

The deleted element is =90

-------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      4

Enter the position of the element to be deleted:        3

The deleted element is =10

-------MENU----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
---------------------
ENTER YOUR CHOICE:      4

Enter the position of the element to be deleted:        8
Invalid Position
```

PROGRAM 2

Design, develop and implement a Program in C for the following operations on Strings

a. **Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)**
b. **Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR**

Support the program with functions for each of the above operations. Don't use Built-in functions.

```
#include<stdio.h>
int flag=0;
void readinput(char str[],char pat[],char rep[])
{
        printf("Enter the Main String\n");
        gets(str);
        printf("Enter the Pattern String\n");
        gets(pat);
        printf("Enter the Replacement String\n");
        gets(rep);
}
void patternmatch(char str[],char pat[],char rep[], char ans[])
{
        int i,j,c,m,k;
        i=j=c=m=k=0;

        while(str[c] != '\0')
        {
                if(str[m] == pat[i])
                {
                        i++;m++;
                        if(pat[i]=='\0')
                        {
                                flag=1;
                                for(k=0;rep[k] != '\0';k++,j++)
                                ans[j] = rep[k];
                                i=0;
                                c=m;
                        }
                }
                else
                {
                        ans[j] = str[c];
                        j++;c++;
                        m=c;
                        i=0;
                }
        }
        ans[j]='\0';
}
```

```
void main()
{
        char str[100],pat[100],rep[100],ans[100];
        readinput(str,pat,rep);
        patternmatch(str,pat,rep,ans);
        if(flag == 0)
        {
                printf("Pattern Not Found\n");
                printf("Resultant String is: %s\n", ans);
        }
        else
        printf("Resultant String is: %s\n", ans);
}
```

OUTPUT:

```
Enter the Main String
hai gat hai cse
Enter the Pattern String
hai
Enter the Replacement String
best
Resultant String is: best gat best cse

Enter the Main String
bset gat best cse
Enter the Pattern String
ise
Enter the Replacement String
ece
Pattern Not Found
Resultant String is: bset gat best cse
```

PROGRAM 3

Design, Develop and Implement a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

a. *Push* **an Element on to Stack**
b. *Pop* **an Element from Stack**
c. **Demonstrate how Stack can be used to check** *Palindrome*
d. **Demonstrate** *Overflow* **and** *Underflow* **situations on Stack**
e. **Display the status of Stack**
f. **Exit**

Support the program with appropriate functions for each of the above operations.

```
#include<stdlib.h>
#include<stdio.h>
#define max_size 5
int stack[max_size],top=-1;
/*Function Prototype*/
void push();
void pop();
void display();
void pali();
int main()
{
        int choice;
        while(1)
        {
                printf("\n\n--------STACK OPERATIONS----------\n");
                printf("1.Push\n");
                printf("2.Pop\n");
                printf("3.Palindrome\n");
                printf("4.Display\n");
                printf("5.Exit\n");
                printf("----------------------");
        printf("\nEnter your choice:\t");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: push(); //to insert an item into stack
                                break;
                        case 2: pop();  //to delete an item from stack
                                break;
                        case 3: pali();  //to check palindrome or not
                                break;
                        case 4: display(); //to display items in stack
                                break;
                        case 5: exit(0);//exit the program
                                break;
                        default:printf("\n Invalid choice:\n");
                                break;
                }
```

```
        }
        return 0;
}
void push() //Inserting element into the stack
{
        int item,n;
        if(top==(max_size-1))
        {
                printf("\nStack Overflow:");
        }
        else
        {
                printf("Enter the element to be inserted:\t");
                scanf("%d",&item);
                top=top+1;
                stack[top]=item;
        }
}
void pop() //deleting an element from the stack
{
        int item;
        if(top==-1)
        {
                printf("Stack Underflow:");
        }
        else
        {
                item=stack[top];
                top=top-1;
                printf("\nThe poped element: %d\t",item);
        }
}
void pali()     //checking whether palindrome or not
{
        int num[10],rev[10],i=0,k,flag=1;
        k=top;
        while(k!=-1)
        {
                num[i++]=stack[k--];
        }
        for(i=0;i<=top;i++)
        {
                if(num[i]==stack[i])
                        continue;
                else
                        flag=0;
        }
```

```
        if(top!=-1)
        {
                if(flag)
                        printf("It is palindrome number\n");
                else
                        printf("It is not a palindrome number\n");
        }
        else
                printf("Stack is Empty:");
}
void display() //Displaying the elements of stack
{
        int i;
        if(top==-1)
        {
                printf("\nStack is Empty:");

        }
        else
        {
                printf("\nThe stack elements are:\n" );
                for(i=top;i>=0;i--)
                {
                        printf("%d\t",stack[i]);
                }
        }
}
```

OUTPUT:

```
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      1
Enter the element to be inserted:       8

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      1
Enter the element to be inserted:       7

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      1
Enter the element to be inserted:       6

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      1

Stack Overflow:
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      4

The stack elements are:
6       7       8       7       6
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      3
It is palindrome number


--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      2

The poped element: 6

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      4

The stack elements are:
7       8       7       6
```

```
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      3
It is not a palindrome number


--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      2

The poped element: 7
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      2

The poped element: 8

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      2

The poped element: 7

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      2

The poped element: 6

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      2
Stack Underflow:
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------
Enter your choice:      5
```

PROGRAM 4

Design, develop and implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %(Remainder), ^(Power) and alphanumeric operands.

```
#include <ctype.h>
#include <stdio.h>
#define SIZE 50 /* Size of Stack */
char s[SIZE];  /* Global declarations */
int top = -1;
push(char elem) /* Function for PUSH operation */
{
        s[++top] = elem;
}
char pop() /* Function for POP operation */
{
        return (s[top--]);
}
int pr(char elem) /* Function for precedence */
{
        switch (elem)
        {
                case '#': return 0;
                case '(': return 1;
                case '+':
                case '-': return 2;
                case '*':
                case '/':
                case '%': return 3;
                case '^': return 4;
        }
}
void main() /* Main Program */
   {
        char infx[50], pofx[50], ch, elem;
        int i = 0, k = 0;
        printf("\n\n enter the Infix Expression : ");
        gets(infx);
        push('#');
        while ((ch = infx[i++]) != '\0')
        {
                if (ch == '(')
                        push(ch);
                else if (isalnum(ch))
                        pofx[k++] = ch;
                else if (ch == ')')
                {
                        while (s[top] != '(')
                                pofx[k++] = pop();
```

```
                elem = pop();     /* Remove ( */
            }
            else    /* Operator */
            {
                while (pr(s[top]) >= pr(ch))
                    pofx[k++] = pop();
                push(ch);
            }
        }
        while (s[top] != '#') /* Pop from stack till empty */
            pofx[k++] = pop();
            pofx[k] = '\0'; /* Make pofx as valid string */
            printf("\n\n Given Infix Expn is: %s\n The Postfix Expn is:%s\n", infx, pofx);
}
```

OUTPUT:

```
enter the Infix Expression : a+(b-c)*d/e^f


Given Infix Expn is: a+(b-c)*d/e^f
The Postfix Expn is:abc-d*ef^/+

enter the Infix Expression : a*(b+c)/d%f$g


Given Infix Expn is: a*(b+c)/d%f$g
The Postfix Expn is:abc+*d/fg$%
```

PROGRAM 5

Design, develop and implement a Program in C for the following Stack Applications

a. **Evaluation of Suffix expression with single digit operands and operators: +, -, *, /,%, ^**
b. **Solving Tower of Hanoi problem with n disks**

Source Code for program 5a:

```
#include<stdio.h>
#include<math.h>
#include<string.h>
int s[30],op1,op2;
int top=-1,i;
char p[30],sym;
int op(int op1,char sym,int op2)
{
        switch(sym)
        {
                case '+':return op1+op2;
                case '-':return op1-op2;
                case '*':return op1*op2;
                case '/':return op1/op2;
                case '%':return op1%op2;
                case '^':
                case '$':return pow(op1,op2);
        }
        return 0;
}
int main()
{
        printf("\nEnter the valid postfix exp:");
        scanf("%s",p);
        for(i=0;i<strlen(p);i++)
        {
                sym=p[i];
                if(sym>='0' && sym<='9')
                        s[++top]=sym-'0';
                else
                {
                        op2=s[top--];
                        op1=s[top--];
                        s[++top]=op(op1,sym,op2);
                }
        }
        printf("\nThe result is %d\n",s[top]);
}
```

OUTPUT:

```
Enter the valid postfix exp: 651-4*23$/+
The result is 8
```

```
Enter the valid postfix exp: 861-*2/32^%
The result is 2
```

Source Code for program 5b:

```
#include<stdio.h>
int count=0,n;
int tower(int n,char s,char t,char d)
{
        if(n==1)
        {
                printf("\n Move disc 1 from %c to %c",s,d);
                count++;
                return 1;
        }
        tower(n-1,s,d,t);
        printf("\n Move disc %d from %c to %c",n,s,d);
        count++;
        tower(n-1,t,s,d);
}
int main( )
{
        printf("\n Enter the no. of discs:");
        scanf("%d",&n);
        tower(n,'A','B','C');
        printf("\n The no. of disc moves is:%d",count);
}
```

OUTPUT:

```
Enter the no. of discs: 1

Move disc 1 from A to C
The no. of disc moves is:1
```

```
Enter the no. of discs:2

Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C
The no. of disc moves is:3
```

```
Enter the no. of discs:3

Move disc 1 from A to C
Move disc 2 from A to B
Move disc 1 from C to B
Move disc 3 from A to C
Move disc 1 from B to A
Move disc 2 from B to C
Move disc 1 from A to C
The no. of disc moves is:7
```

```
Enter the no. of discs:4

Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C
Move disc 3 from A to B
Move disc 1 from C to A
Move disc 2 from C to B
Move disc 1 from A to B
Move disc 4 from A to C
Move disc 1 from B to C
Move disc 2 from B to A
Move disc 1 from C to A
Move disc 3 from B to C
Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C
The no. of disc moves is:15
```

PROGRAM 6

Design, develop and implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. **Insert an Element on to Circular QUEUE**
b. **Delete an Element from Circular QUEUE**
c. **Demonstrate *Overflow* and *Underflow* situations on Circular QUEUE**
d. **Display the status of Circular QUEUE**
e. **Exit**

Support the program with appropriate functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 5
char q[SIZE];
int i,r=-1;
int option,f=0;
int j,count=0;
void insert()
{
        if(count==SIZE)
        printf("\n Q is Full\n");
        else
        {
                 r=(r+1)%SIZE;
                printf("\nEnter the item:");
                scanf(" %c",&q[r]);
                count++;
        }
}
void delet()
{
        if(count==0)
         printf("\nQ is empty\n");
         else
         {
                printf("\nDeleted item is: %c",q[f]);
                count--;
                 f=(f+1)%SIZE;
         }
}
void display()
{
        if(count==0)
         printf("\nQ is Empty\n");
        else
         {
                 i=f;
                for(j=1;j<=count;j++)
                {
```

```
                printf(" %c",q[i]);
                i=(i+1)%SIZE;
            }
        }
}
int main()
{
    for(;;)
    {
        printf("\n1.Insert  2.Delete\n 3.Display 4.Exit");
        printf("\nEnter your option:");
        scanf("%d",&option);
        switch(option)
        {
            case 1: insert();//Inserting items to Queue
                break;
            case 2: delet();//Deleting items from Queue
            break;
            case 3:display();       //Displaying items from Queue
             break;
            default:exit(0);
        }
    }
}
```

OUTPUT:

```
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:2

Q is empty

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:3

Q is Empty
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: A

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: B

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: C

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: D

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: E

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

 Q is Full
```

```
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:3
A B C D E
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:2

Deleted item is: A
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:2

Deleted item is: B
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:2

Deleted item is: C
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:3
 D E
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: A

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: B

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:3
 D E A B
1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

Enter the item: C

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:1

 Q is Full

1.Insert  2.Delete  3.Display 4.Exit
Enter your option:4
```

PROGRAM 7

Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Branch, Sem, PhNo*

a. **Create a SLL of N Students Data by using** *front insertion*.
b. **Display the status of SLL and count the number of nodes in it**
c. **Perform Insertion and Deletion at End of SLL**
d. **Perform Insertion and Deletion at Front of SLL (Demonstration of stack)**
e. **Exit**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct student
{
        char usn[11];
        char name[25];
        int sem;
        char branch[5];
        unsigned long long phno;
};
typedef struct student STUD;

struct node
{   char usn[11];
        char name[25];
        int sem;
        char branch[5];
        unsigned long long phno;
        struct node *next;
};
typedef struct node NODE;

NODE *first;
NODE* copyNode(STUD s)
{       NODE * temp;
        temp= (NODE *)malloc(sizeof(NODE));
         if(temp==NULL)
        {
                printf("Memory cannot be allocated\n");
        }
        else
        {   strcpy(temp->usn,s.usn);
                strcpy(temp->name, s.name);
                strcpy(temp->branch, s.branch);
                temp->sem=s.sem;
                temp->phno=s.phno;
                temp->next=NULL;
                return temp;
}       }
```

```
void addrear(STUD s)
{
        NODE *temp,*cur;
        temp=copyNode(s) ;
        if(first==NULL)
        {
                first=temp; return;
        }
        cur=first;
        while(cur->next != NULL)
        {
                 cur=cur->next;
        }
        cur->next =temp;
        return ;
}

void addfront(STUD s)
{
         NODE *temp;
        temp=copyNode(s); //ssn,name, dept, design,salary, pno);
        if (first== NULL)
{
                first=temp;
}
        else
        {
                temp->next=first;
                first=temp;
        }
         return ;
}

void display(NODE *temp)
{
        printf("%s \t", temp->usn);
        printf("%s \t", temp->name);
        printf("%s \t", temp->branch);
        printf("%d \t",temp->sem);
        printf("%llu \n", temp->phno);
}

void deletefront()
{
        NODE *temp; int num;
        temp=first;
        if(first==NULL)
```

```
        {
                printf("List is Empty"); return;
        }
        if(first->next==NULL)
                first=NULL;
        else
        {
                first=first->next;
         }
        printf("Deleted Node is:\n");
         display(temp);
        free(temp);
        return ;
}

void deleterear()
{
        NODE *cur, *prev;
        cur=first;
        prev=NULL;
         if(first==NULL)
        {
                printf("List is Empty"); return;
        }
        if(first->next==NULL)
        {
                display(cur);
                first=NULL;
                free(cur); return;
        }

        while(cur->next!=NULL)
        {
                prev=cur;
                cur=cur->next;
        }
        prev->next=NULL;
        printf("Deleted Node is:\n");
        display(cur);
        free(cur);
        return;
}

void  displayList()
{
        NODE *r;
        r=first;
        printf("USN\tName\tBrh\tSem\tPhone\n");
```

```
        if(r==NULL)
        return;
        while(r!=NULL)
        {
                display(r);
                r=r->next;
        }
        printf("\n");
}

STUD input()
{
        STUD s;
        printf("Enter USN   : ");
        scanf("%s",s.usn);
        printf("Enter Name  : ");
        scanf("%s",s.name);
        printf("Enter Branch: ");
        scanf("%s",s.branch);
         printf("Enter Sem:");
        scanf("%d",&s.sem);
        printf("Enter Phone no : ");
         scanf("%llu",&s.phno);
         return s;
}

int count()
{
   NODE *n;
   int c=0;
   n=first;
   while(n!=NULL)
   {    n=n->next;
        c++;
   }
   return c;
}

int  main()
{  STUD s;   int i,  ch, n;
   first=NULL;
   while(1)
   {
        printf("\nList Operations\n");
        printf("==========\n");
        printf("1.Create List of n students by using front Insert\n");
        printf("2.Display the status and count the nodes\n");
        printf("3.Perform Insertion and Deletion at End of SLL\n");
```

```
printf("4.Perform Insertion and Deletion at Front of SLL\n");
printf("5.Exit\n");
printf("Enter your choice : ");
scanf("%d",&i);
switch(i)
{
        case 1 : printf("Enter the number of students\n");
                 scanf("%d",&n);
                 for(i=1;i<=n;i++)
                 {
                         printf("\nEnter the details of student %d\n",i);
                         s=input();
                         addfront(s);
                 }
                 break;

        case 2 : if(first==NULL)
                 {
                          printf("List is Empty\n");
                 }
                 else
                 {
                         printf(" Node Count=%d\t & Elements  in the list are : \n",
                         count());
                         displayList();
        }
                 break;

        case 3 : printf(" 1. Insert at End and 2 Delete From End=");
                 scanf("%d",&ch);
                 if(ch==1)
                 {
                         s=input();
                         addrear(s);
                 }
                 else if(ch==2)
        deleterear();
                  else
                          printf(" Sorry wrong operation\n");
                 break;

        case 4 : printf(" 1. Insert at Front and 2  Delete From Front=");
         scanf("%d",&ch);
                 if(ch==1)
                 {
                         s=input();
                         addfront(s);
                 }
```

```
                else if(ch==2)
            deletefront();
                else
                printf(" Sorry wrong operation\n");
                 break;

            case 5:  exit (0);
            default : printf("Invalid option\n");
        }
    }
    return 0;
}
```

OUTPUT:

```
List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 1
Enter the number of students
2

Enter the details of student 1
Enter USN   : 001
Enter Name  : shakuntaladevi
Enter Branch: cse
Enter Sem:3
Enter Phone no : 1111111111

Enter the details of student 2
Enter USN   : 002
Enter Name  : indranooyi
Enter Branch: cse
Enter Sem:3
Enter Phone no : 2222222222

List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 2
 Node Count=2    & Elements  in the list are :
USN     Name    Brh     Sem     Phone
002     indranooyi      cse     3       2222222222
001     shakuntaladevi  cse     3       1111111111
```

```
List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 3
 1. Insert at End and 2 Delete From End=1
Enter USN   : 003
Enter Name  : chandakochhar
Enter Branch: cse
Enter Sem:3
Enter Phone no : 3333333333

List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 2
 Node Count=3    & Elements  in the list are :
USN     Name    Brh     Sem     Phone
002     indranooyi      cse     3       2222222222
001     shakuntaladevi  cse     3       1111111111
003     chandakochhar   cse     3       3333333333
```

```
List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 4
 1. Insert at Front and 2  Delete From Front=1
Enter USN   : 004
Enter Name  : shikhasharma
Enter Branch: cse
Enter Sem:3
Enter Phone no : 4444444444

List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 2
 Node Count=4    & Elements  in the list are :
USN     Name    Brh     Sem     Phone
004     shikhasharma    cse     3       4444444444
002     indranooyi      cse     3       2222222222
001     shakuntaladevi  cse     3       1111111111
003     chandakochhar   cse     3       3333333333
```

```
List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 4
 1. Insert at Front and 2  Delete From Front=2
Deleted Node is:
004     shikhasharma    cse     3       4444444444
```

```
List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 2
 Node Count=3    & Elements  in the list are :
USN     Name    Brh     Sem     Phone
002     indranooyi      cse     3       2222222222
001     shakuntaladevi  cse     3       1111111111
003     chandakochhar   cse     3       3333333333


List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 3
 1. Insert at End and 2 Delete From End=2
Deleted Node is:
003     chandakochhar   cse     3       3333333333

List Operations
==============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 2
 Node Count=2    & Elements  in the list are :
USN     Name    Brh     Sem     Phone
002     indranooyi      cse     3       2222222222
001     shakuntaladevi  cse     3       1111111111
```

```
List Operations
==============
1.Create List of n students by using front Inse
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of Sl
5.Exit
Enter your choice : 5
```

PROGRAM 8

Design, develop and implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN,Name, Dept, Designation, Sal, PhNo*

a. **Create a DLL of N Employees Data by using** *end insertion*.
b. **Display the status of DLL and count the number of nodes in it**
c. **Perform Insertion and Deletion at End of DLL**
d. **Perform Insertion and Deletion at Front of DLL**
e. **Demonstrate how this DLL can be used as Double Ended Queue**
f. **Exit**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct employee
{
        char ssn[11];
        char name[21];
        char dept[15];
         char design[15];
        int  salary;
        unsigned long long  phno;
};
typedef struct employee EMP;

struct node
{
         char ssn[11];
         char name[21];
          char dept[15];
        char design[15];
        int  salary;
        unsigned long long  phno;
        struct node *next;
        struct node *prev;
};
typedef struct node NODE;
NODE *first;
NODE* copyNode(EMP e)
{
        NODE * temp;
        temp= (NODE *)malloc(sizeof(NODE));
         if(temp==NULL)
        {
                printf("Memory cannot be allocated\n");
        }
        else
        {
                 strcpy(temp->ssn,e.ssn);
```

```
                strcpy(temp->name, e.name);
                strcpy(temp->dept, e.dept);
                strcpy(temp->design, e.design);
                temp->salary=e.salary;
                temp->phno=e.phno;
                temp->next=NULL;
                temp->prev=NULL;
                return temp;
        }
}

void addrear(EMP e)
{
        NODE *temp,*cur, *prev;
         temp=copyNode(e) ;
        if(first==NULL)
        {
                first=temp; return;
        }
        cur=first; prev=NULL;
        while(cur->next != NULL)
        {
                prev=cur;
                cur=cur->next;
        }
        cur->next =temp;
        temp->prev=cur;
        return ;
}

void addfront(EMP e)
{
        NODE *temp;
        temp=copyNode(e);
        if (first== NULL)
        {
                first=temp;
        }
        else
        {
                 temp->next=first;
                first->prev=temp;
                first=temp;
        }
         return ;
}
```

```
void display(NODE *r)
{
        printf("%s\t", r->ssn);
        printf("%s\t", r->name);
        printf("%s\t", r->dept);
        printf("%s\t",r->design);
        printf("%d\t",r->salary);
        printf("%llu\n", r->phno);
}

void deletefront()
{
        NODE *temp; int num;
        temp=first;
        if(first==NULL)
        {
                printf(" List is Empty \n");return ;
        }
        if(first->next==NULL)
           first=NULL;
        else
        {       first=first->next;
                first->prev=NULL;
        }
        printf("SSN\tName\tDept\tDesignation\tSalary\tPhone\n");
        display(temp);
        free(temp);
        return ;
}

void deleterear()
{
        NODE *cur, *prev;
        cur=first;
        prev=NULL;
        if(first==NULL)
        {
                printf(" List is Empty \n");return ;
        }
        if(first->next==NULL)
        {
                first=NULL;
        }
        else
        {
                while(cur->next!=NULL)
                {
                        prev=cur;
```

```
                cur=cur->next;
            }
            prev->next=NULL;
        }
        printf("SSN\tName\tDept\tDesignation\tSalary\tPhone\n");
        display(cur);
        free(cur);
        return;
}

void  displayList()
{
        NODE *r;
        r=first;
        if(r==NULL)
        {
            return;
         }
        printf("SSN\t Name\t Dept\t Designation\t salary\tPhone\n");
        while(r!=NULL)
        {
             display(r);
            r=r->next;
         }
        printf("\n");
}

int count()
{
   NODE *n;
   int c=0;
   n=first;
   while(n!=NULL)
   {
        n=n->next;
        c++;
   }
   return c;
}

EMP input()
{
        EMP e;
        printf("Enter SSN   : ");
        scanf("%s",e.ssn);
        printf("Enter Name  : ");
        scanf("%s",e.name);
        printf("Enter dept: ");
```

```
        scanf("%s",e.dept);
        printf("Enter Designation :");
        scanf("%s",e.design);
        printf("Enter Salary:");
        scanf("%d",&e.salary);
        printf("Enter Phone no : ");
        scanf("%llu",&e.phno);
        return e;
}

int  main()
{
        EMP e;
        int i, ch, n;
        first=NULL;
        while(1)
        {
                printf("\nList Operations\n");
                printf("================\n");
                printf("1.Create a DLL of N Employees Data by using end insertion\n");
                printf("2.Display the status of DLL and count the number of nodes in it\n");
                printf("3.Perform Insertion and Deletion at End of DLL\n");
                printf("4.Perform Insertion and Deletion at Front of DLL\n");
                printf("5.Demonstration of this DLL as Double Ended Queue\n");
                printf("6.Exit\n");
                printf("Enter your choice : ");
                scanf("%d",&i);
                switch(i)
                 {
                        case 1 : printf("Enter the number of Employees\n");
                                 scanf("%d",&n);
                                 for(i=1;i<=n;i++)
                                 {
                                        printf("\nEnter the details of Employee %d\n",i);
                                        e=input();
                                 addrear(e);
                                }
                                 break;
                        case 2 : if(first==NULL)
                                 {
                                        printf("List is Empty\n");
                                 }
                                 else
                                 {
                                 printf(" Node Count=%d\t & Elements  in the list are : \n", count());
                                 displayList();
                                 }
                                 break;
```

```
                case 3: printf(" 1. Insert at End and 2  Delete From End=");
                scanf("%d",&ch);
                if(ch==1)
                        {
                                e=input();
                                addrear(e);
                        }
                        else if(ch==2)
                                deleterear();
                        else
                                 printf(" Sorry wrong operation\n");
                 break;


                case 4:  printf(" 1. Insert at Front and 2  Delete From Front=");
                        scanf("%d",&ch);
                        if(ch==1)
                        {
                                e=input();
                                addfront(e);
                        }
                        else if(ch==2)
                        deletefront();
                 else
            printf(" Sorry wrong operation\n");
                        break;

                        case 5: printf("This DLL can be used as Double Ended Queue by inserting
and deleting from both ends \n");
                                break;

                        case 6  :       return 0;
                        default :       printf("Invalid option\n");
                }
        }
        return 0;
}
```

OUTPUT:

```
List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 1
Enter the number of Employees
3

Enter the details of Employee 1
Enter SSN   : 111
Enter Name  : ram
Enter dept: sales
Enter Designation :manager
Enter Salary:50000
Enter Phone no : 9999999999

Enter the details of Employee 2
Enter SSN   : 222
Enter Name  : sham
Enter dept: design
Enter Designation :manager
Enter Salary:55000
Enter Phone no : 8888888888

Enter the details of Employee 3
Enter SSN   : 3
Enter Name  : sunny
Enter dept: market
Enter Designation :manager
Enter Salary:60000
Enter Phone no : 7777777777

List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 2
 Node Count=3    & Elements  in the list are :
SSN      Name    Dept    Designation     salary Phone
111      ram     sales   manager 50000   9999999999
222      sham    design  manager 55000   8888888888
3        sunny   market  manager 60000   7777777777


List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 3
 1. Insert at End and 2  Delete From End=1
Enter SSN   : 444
Enter Name  : jack
Enter dept: sales
Enter Designation :admin
Enter Salary:45000
Enter Phone no : 6666666666

List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 4
 1. Insert at Front and 2  Delete From Front=1
Enter SSN   : 555
Enter Name  : rose
Enter dept: admin
Enter Designation :manager
Enter Salary:50000
Enter Phone no : 5555555555
```

```
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 2
 Node Count=5   & Elements  in the list are :
SSN      Name   Dept    Designation     salary Phone
555      rose   admin   manager 50000   5555555555
111      ram    sales   manager 50000   9999999999
222      sham   design  manager 55000   8888888888
3        sunny  market  manager 60000   7777777777
444      jack   sales   admin   45000   6666666666


List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 3
 1. Insert at End and 2  Delete From End=2
SSN     Name    Dept    Designation     Salary  Phone
444     jack    sales   admin   45000   6666666666

List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 4
 1. Insert at Front and 2  Delete From Front=2
SSN     Name    Dept    Designation     Salary  Phone
```

```
List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 2
 Node Count=3   & Elements  in the list are :
SSN      Name   Dept    Designation     salary Phone
111      ram    sales   manager 50000   9999999999
222      sham   design  manager 55000   8888888888
3        sunny  market  manager 60000   7777777777


List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 3
 1. Insert at End and 2  Delete From End=2
SSN     Name    Dept    Designation     Salary  Phone
3       sunny   market  manager 60000   7777777777

List Operations
==============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 4
 1. Insert at Front and 2  Delete From Front=2
SSN     Name    Dept    Designation     Salary  Phone
```

```
SSN      Name     Dept     Designation      Salary  Phone
222      sham     design   manager 55000    8888888888

List Operations
===============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 4
 1. Insert at Front and 2  Delete From Front=2
 List is Empty

List Operations
===============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 3
 1. Insert at End and 2  Delete From End=2
 List is Empty

List Operations
===============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 2
List is Empty

List Operations
===============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 5
This DLL can be used as Double Ended Queue by inserting and deleting from both ends

List Operations
===============
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 6
```

PROGRAM 9

Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes

a. **Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$**
b. **Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)**

Support the program with appropriate functions for each of the above operations.

Program 9a:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int coef,px,py,pz, x,y,z,i;
int val;

struct node
{
        int coef,px, py,pz;
        struct node *next;
};
typedef struct node NODE;
NODE *first;

void insert(int coef,int px, int py, int pz)
{
        NODE *temp,*cur;
        temp= (NODE *)malloc(sizeof(NODE));
        temp->coef=coef; temp->px=px; temp->py=py; temp->pz=pz;
        if(first==NULL)
        {
                temp->next=temp;
                first=temp; return;
        }
        if(first->next==first)
        {
                first->next=temp; temp->next=first;
        }
        cur=first;
        while(cur->next!=first)
        {
                cur=cur->next;
        }
        cur->next=temp;
        temp->next=first;
        return;
}
```

```
void display()
{
        NODE *cur;
        if(first==NULL)
        {
                printf("List is empty\n");return;
        }
        cur=first;
        while(cur->next!=first)
        {
                printf("%d ",cur->coef);
                printf(" x^%d",cur->px);
                printf(" y^%d",cur->py);
                printf(" z^%d + ",cur->pz);
                cur=cur->next;
        }
        printf("%d ",cur->coef);
        printf(" x^%d",cur->px);
       printf(" y^%d",cur->py);
       printf(" z^%d\n",cur->pz);
        return;
}

int  evaluate(int x, int y, int z)
{
        NODE *cur; int v,s=0, v1,v2,v3;
        if(first==NULL)
        {
                printf("List is empty\n");return 0;
        }
        cur=first;
        while(cur->next!=first)
        {
                v=cur->coef*pow(x, cur->px)*pow(y, cur->py)*pow(z,cur->pz);
                s=s+v;
                cur=cur->next;
        }
        v=cur->coef*pow(x, cur->px)*pow(y, cur->py)*pow(z,cur->pz);
        s=s+v;
        return s;
}

int main()
{
        int coef,px,py,pz, x,y,z,i;
        int val;
        first=NULL;
        while(1)
```

```
    {
        printf("1. Insert polynomial at end\n");
        printf("2. Display\n");
        printf("3. Evaluate\n");
        printf("4. Exit\n");
        printf("Enter Choice= \t");
        scanf("%d",&i);
        switch(i)
        {
            case 1 : printf("Enter Coefficient= \t");
                     scanf("%d",&coef);
                     printf("Enter powers of x y z values= \t");
                     scanf("%d%d%d",&px, &py,&pz);
                     insert(coef,px,py,pz);
                     break;

            case 2 : display();
                     break;

            case 3 : printf("\n Enter x y & z values for evaluation: \t");
                     scanf("%d%d%d",&x,&y,&z);
                     val=evaluate(x,y,z);
                     printf("\nValue=%d\n",val);
                     break;

            case 4 :  return 0;

            default : printf(" Wrong choice. Enter 1,2 3\n"); break;
        }
    }
}
```

OUTPUT:

```
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=   1
Enter Coefficient=      4
Enter powers of x y z values=   3 3 2
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=   1
Enter Coefficient=      5
Enter powers of x y z values=   3 2 2
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=   1
Enter Coefficient=      3
Enter powers of x y z values=   2 2 2
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=   1
Enter Coefficient=      6
Enter powers of x y z values=   1 1 1
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=   2
4  x^3 y^3 z^2 + 5  x^3 y^2 z^2 + 3  x^2 y^2 z^2 + 6  x^1 y^1 z^1
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=   3

 Enter x y & z values for evaluation:   3 2 1

Value=1548
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=   4
```

Program 9b:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

struct node      //Defining Polynomial fields
{
int coef, px, py, pz,flag;
struct node *link;
};

typedef struct node * NODE;
insert(NODE head,int cof,int x,int y, int z) //inserting term to poly
{
        NODE cur,tmp;
        tmp= (NODE)malloc(sizeof(struct node));    //Allocates memory
        int cf,px,py,pz;
        cur=head->link;
        tmp->coef=cof;
        tmp->px=x;
        tmp->py=y;
        tmp->pz=z;
        tmp->flag=0;
        while(cur->link!=head)          //Identifying last node
                cur=cur->link;
                cur->link=tmp;
                tmp->link=head;
}

NODE create_list(NODE head)         //For creating poly1 & poly2
{
        int i,n,cf,px,py,pz;
        printf("Enter the number of terms : ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                printf("Enter the Co-ef, px, py, pz : ");
                scanf("%d %d %d %d",&cf,&px,&py,&pz);
                insert(head,cf,px,py,pz);
        }
        return head;
}/*End of create_list()*/


NODE add_poly(NODE h1,NODE h2,NODE h3)
{
        NODE cur1,cur2;
        int scf;
```

```
        cur1=h1->link;
        cur2=h2->link;
        while(cur1 != h1)       //Till end of poly1
        {
                if(cur2 == h2)
                        cur2=h2->link;
                while(cur2 != h2)       //Till end of poly2
                {
                if(cur1->px == cur2->px && cur1->py == cur2->py && cur1->pz == cur2->pz)
                {       //Add & insert if co-ef's of both poly is equal
                                scf = cur1->coef + cur2->coef;
                                insert(h3,scf,cur1->px,cur1->py,cur1->pz);
                                cur2->flag=1;
                                cur2=h2->link;
                                break;
                }
                cur2=cur2->link;
                }
                if(cur1 == h1)
                        break;
                if(cur2 == h2) //If co-ef of poly1 is not matched, insert it to poly3
                insert(h3,cur1->coef,cur1->px,cur1->py,cur1->pz);
                cur1=cur1->link;
        }
        cur2=h2->link;
        while(cur2 != h2)       //remaining poly2 nodes inserted to poly3
        {
                if(cur2->flag==0)
                        insert(h3,cur2->coef,cur2->px,cur2->py,cur2->pz);
                cur2=cur2->link;
        }
        return h3;
}

void display(NODE head)
{
   NODE cur;
        if(head->link==head) //if poly is empty
        {
                printf("List is empty\n");
                return;
        }
        cur=head->link;
        while(cur != head)      //display all terms till end
        {
                if(cur->coef > 0)
                        printf(" +%dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);
                else if (cur->coef < 0)
```

```
                printf(" %dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);
                cur=cur->link;
    }
printf("\n");
}/*End of display() */

void main()
{
        int choice,data,item,pos;
   NODE head1,head2,head3;

        head1=(NODE)malloc(sizeof(struct node));
        head1->link=head1;    //poly1

        head2=(NODE)malloc(sizeof(struct node));
        head2->link=head2;    //poly2

        head3=(NODE)malloc(sizeof(struct node));
        head3->link=head3;    //poly3

        printf("\n1.Create Polynomial 1\n");
        head1=create_list(head1);

        printf("\n2.Create Polynomial 2\n");
        head2=create_list(head2);
        printf("\nPolynomial 1 is :");
        display(head1);

        printf("\nPolynomial 2 is :");
        display(head2);

        head3=add_poly(head1,head2,head3);          //Add both polynomials

        printf("\nAddition of two Polynomial is :");
        display(head3);
}
```

OUTPUT:

```
admin1@global:~$ ./a.out

1.Create Polynomial 1
Enter the number of terms : 2
Enter the Co-ef, px, py, pz : 2
3
4
3
Enter the Co-ef, px, py, pz : 1
3
4
3

2.Create Polynomial 2
Enter the number of terms : 2
Enter the Co-ef, px, py, pz : 3
1
2
3
Enter the Co-ef, px, py, pz : 4
2
3
1

Polynomial 1 is : +2x^3y^4z^3  +1x^3y^4z^3

Polynomial 2 is : +3x^1y^2z^3  +4x^2y^3z^1

Addition of two Polynomial is : +2x^3y^4z^3  +1x^3y^4z^3  +3x^1y^2z^3  +4x^2y^3z^1
```

PROGRAM 10

Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers

a. **Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2**
b. **Traverse the BST in In-order, Preorder and Post Order**
c. **Search the BST for a given element (KEY) and report the appropriate message**
d. **Exit**

```
#include<stdio.h>
#include<stdlib.h>
int choice,data,key;
struct node
{
int info;
struct node *lchild,*rchild;
};
typedef struct node *NODE;

int main()
{
NODE root=NULL;
NODE CREATE(NODE,int);
void INORDER(NODE),POSTORDER(NODE),PREORDER(NODE);
NODE SEARCH_NODE(NODE,int);

while(1)
{
printf("\n1:CREATE\n2:TREE TRAVERSAL\n3.SEARCH\n4.EXIT");
printf("\nEnter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("\nEnter data to be inserted\n");
scanf("%d",&data);
root=CREATE(root,data);
break;

case 2: if(root==NULL)
printf("\nEMPTY TREE\n");
else
{
printf("\nThe Inorder display : ");
INORDER(root);
printf("\nThe Preorder display : ");
PREORDER(root);
printf("\nThe Postorder display : ");
POSTORDER(root);
}
break;
```

```
case 3: printf("\nenter the key to search:\n");
scanf("%d",&key);
SEARCH_NODE(root,key);
break;

case 4:  exit(0);
}
}
}

NODE  CREATE(NODE root,int data)
{
NODE newnode,x,parent;
newnode=(NODE)malloc(sizeof(struct node));
newnode->lchild=newnode->rchild=NULL;
newnode->info=data;

if(root==NULL)
root=newnode;
else
{
x=root;
while(x!=NULL)
{
parent=x;
if(x->info<data)
x=x->rchild;
else if(x->info>data)
x=x->lchild;
else
{
printf("\nNode is already present in the tree\n");
return(root);
}
}
if(parent->info<data)
parent->rchild=newnode;
else
parent->lchild=newnode;
}
return(root);
}

void INORDER(NODE root)
{
if(root!=NULL)
{
INORDER(root->lchild);
```

```
printf("%d ",root->info);
INORDER(root->rchild);
}
}

void PREORDER(NODE root)
{
if(root!=NULL)
        {
printf("%d ",root->info);
PREORDER(root->lchild);
PREORDER(root->rchild);
        }
}

void POSTORDER(NODE root)
{
if(root!=NULL)
{
POSTORDER(root->lchild);
POSTORDER(root->rchild);
printf("%d ",root->info);
}
}

NODE SEARCH_NODE(NODE root, int key)
{
NODE cur,q,parent,successor;
if(root==NULL)
{
printf("\nTree is empty\n");
return root;
}
parent=NULL,cur=root;
while(cur!=NULL)
{
if(key==cur->info)
break;
parent=cur;
cur= (key<cur->info)?cur->lchild:cur->rchild;
}
if(cur==NULL)
{
printf("\nData is not found\n");
return root;
}
printf("\nData %d is found\n",key);
}
```

OUTPUT:

```
1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
2

EMPTY TREE

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
6
1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
9

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
5

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
2

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
8
```

```
1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
15

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
24

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
14

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
7

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
8

Node is already present in the tree
```

```
1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
5

Node is already present in the tree

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
2

Node is already present in the tree

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
2

The Inorder display : 2 5 6 7 8 9 14 15 24
The Preorder display : 6 5 2 9 8 7 15 14 24
The Postorder display : 2 5 7 8 14 24 15 9 6
1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
3

enter the key to search:
14

Data 14 is found

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
3

enter the key to search:
4

Data is not found
```

PROGRAM 11

Design, develop and implement a Program in C for the following operations on Graph (G) of Cities

a. **Create a Graph of N cities using Adjacency Matrix.**
b. **Print all the nodes reachable from a given starting node in a digraph using BFS method**

```
#include<stdio.h>
#include<stdlib.h>
int n,a[10][10],i,j,source,s[10],choice,count;
void bfs(int n,int a[10][10],int source,int s[]) //BFS Algorithm
{
        int q[10],u;
        int front=1,rear=1;
        s[source]=1;
        q[rear]=source;
        while(front<=rear)
        {
                u=q[front];
                front=front+1;
                for(i=1;i<=n;i++)
                if(a[u][i]==1 &&s[i]==0)
                {
                        rear=rear+1;
                        q[rear]=i;
                        s[i]=1;
                }
        }
}
int main()
{
        printf("Enter the number of nodes : ");
        scanf("%d",&n);
        printf("\n Enter the adjacency matrix\n");
        for(i=1;i<=n;i++)       //Provide matrix of 0's and 1's
                for(j=1;j<=n;j++)
                        scanf("%d",&a[i][j]);
        while(1)
        {
                printf("\nEnter your choice\n");
                printf("1.BFS\n 2.Exit\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("\n Enter the source :");
                                scanf("%d",&source); //Provide source for BFS
                                for(i=1;i<=n;i++)
                                s[i]=0;
```

```
                        bfs(n,a,source,s);
                        for(i=1;i<=n;i++)
                {
                        if(s[i]==0)
                                printf("\n The node %d is not reachable",i);
                        else
                                printf("\n The node %d is reachable",i);
                }
                        break;
                case 2: exit(0);
        }
    }
}
```

OUTPUT:

```
Enter the number of nodes : 5

 Enter the adjacency matrix
0 1 0 1 0
0 0 1 0 0
1 0 0 0 0
0 0 1 0 0
0 0 1 1 0

Enter your choice
1.BFS
 2.Exit
1

 Enter the source :5

 The node 1 is reachable
 The node 2 is reachable
 The node 3 is reachable
 The node 4 is reachable
 The node 5 is reachable
Enter your choice
1.BFS
 2.Exit
1

 Enter the source :1

 The node 1 is reachable
 The node 2 is reachable
 The node 3 is reachable
 The node 4 is reachable
 The node 5 is not reachable
Enter your choice
1.BFS
 2.Exit
```

//Print all the nodes reachable from a given starting node in a digraph using DFS method

Source Code:

```
#include<stdio.h>
 void dfs(int n,int a[10][10],int source,int s[])
 {
   int i;
   s[source]=1;
    for(i=1;i<=n;i++)
    if(a[source][i]==1 && s[i]==0)
    dfs(n,a,i,s);
 }
 int main()
 {
        int i,j,source,s[10];
        int n,a[10][10];
        int count=0;
        printf("\nEnter the number of nodes : ");
        scanf("%d",&n);
        printf("\nEnter the paths \n");
        for(i=1;i<=n;i++)
           for(j=1;j<=n;j++)
                scanf("%d",&a[i][j]);

        printf("\nEnter the source vertex :");
        scanf("%d",&source);
        for(i=1;i<=n;i++)
                 s[i]=0;
        dfs(n,a,source,s);
         for(i=1;i<=n;i++)
          {
           if(s[i]==0)
              printf("\n The node %d is not reachable",i);
           else
              printf("\n The node %d is reachable",i);
           }
   }
```

PROGRAM 12

Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers.

Design and develop a Program in C that uses Hash function H: K->L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing

```
#include<stdio.h>
#define MAX_ADDR 5

struct employee
{
        int emp_id, emp_age;
        char emp_name[25];
}emp[MAX_ADDR];   //Defining Array of Structures

void main()
{
        int i, ch, count = 0, index, haddr, id, flag = 0;
        for(;;)
        {
                printf("Enter 1 to insert record \nEnter 2 to search  record\nEnter 3 to Exit\n");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1: if(count == MAX_ADDR) //When Records is full
                                {
                                        printf("No free address space\n");
                                        break;
                                }
                                printf("Enter employee id\n");
                                scanf("%d", &id);
                                haddr = hash(id);
                                printf("Home address is %d\n", haddr);
                                for(i=0; i<MAX_ADDR; i++)
                                {
                                        index = (haddr+i)%MAX_ADDR;//For Wrap Around
                                        if(emp[index].emp_id == 0)
                                        {
                                                emp[index].emp_id = id;
                                                printf("Enter the employee name\n");
                                                scanf("%s", emp[index].emp_name);
                                                printf("Enter the employee age\n");
                                                scanf("%d", &emp[index].emp_age);
                                                count++;
                                        printf("Successfuly inserted at Actual Address%d:\n\n", index);
                                                break;
                                        }
                                }
```

```
                        break;
                case 2: printf("Enter employee id to be searched?\n");
                        scanf("%d",&id);
                        haddr = hash(id);
                        for(i=0; i<MAX_ADDR; i++)
                        {
                                index = (haddr+i)%MAX_ADDR;
                                if(emp[index].emp_id == 0)
                                {
                                        flag = 1;
                                        break;
                                }
                                else if(emp[index].emp_id == id)
                                {
                                printf("Employee id is%d\n", emp[index].emp_id);
                                printf("Employee name is %s\n", emp[index].emp_name);
                                printf("Employee age is %d\n", emp[index].emp_age);
                                printf("Search Length is: %d\n", ++i);
                                break;
                                }
                        }
                        if(flag == 1 || i == MAX_ADDR)
                        {
                                printf("Key not present\n");
                        }
                        break;
                case 3:  exit(0);

                        default: printf("Invalid Choice..\n");
                }
        }
}

int hash (int key)          //Generates Hash address or Home address
{
        return key % MAX_ADDR;
}
```

OUTPUT:

```
Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
1
Enter employee id
123
Home address is 3
Enter the employee name
sunny
Enter the employee age
26
Successfuly inserted at Actual Address 3:

Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
1
Enter employee id
125
Home address is 0
Enter the employee name
ram
Enter the employee age
25
Successfuly inserted at Actual Address 0:

Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
1
Enter employee id
124
Home address is 4
Enter the employee name
sham
Enter the employee age
27
Successfuly inserted at Actual Address 4:

Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
1
Enter employee id
120
Home address is 0
Enter the employee name
jack
Enter the employee age
28
Successfuly inserted at Actual Address 1

Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
1
Enter employee id
121
Home address is 1
Enter the employee name
rose
Enter the employee age
23
Successfuly inserted at Actual Address 2

Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
1
No free address space
Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
2
Enter employee id to be searched?
121
Employee id is121
Employee name is rose
Employee age is 23
Search Length is: 2
Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
2
Enter employee id to be searched?
125
Employee id is125
Employee name is ram
Employee age is 25
Search Length is: 1
Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
2
Enter employee id to be searched?
123
Employee id is123
Employee name is sunny
Employee age is 26
Search Length is: 1
Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
2
Enter employee id to be searched
234
Key not present
Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
2
Enter employee id to be searched
128
Key not present
Enter 1 to insert record
Enter 2 to search record
Enter 3 to Exit
```